Indonesian-English Textual Similarity Detection Using Universal Sentence Encoder (USE) and Facebook AI Similarity Search (FAISS)

Lucia D. Krisnawati[1*], Aditya W. Mahastama[2], Su-Cheng Haw[3],
Kok-Why Ng[4], and Palanichamy Naveen[5]
[1,2]Informatics Department, Faculty of Information Technology, Universitas Kristen Duta Wacana
Yogyakarta, Indonesia 55224
[3−5]Faculty of Computing and Informatics, Multimedia University
Cyberjaya, Malaysia 63100
Email: [1]krisna@staff.ukdw.ac.id, [2]mahas@staff.ukdw.ac.id, [3]sucheng@mmu.edu.my,
[4]kwng@mmu.edu.my, [5]p.naveen@mmu.edu.my

***Abstract*—The tremendous development in Natural Language Processing (NLP) has enabled the detection of bilingual and multilingual textual similarity. One of the main challenges of the Textual Similarity Detection (TSD) system lies in learning effective text representation. The research focuses on identifying similar texts between Indonesian and English across a broad range of semantic similarity spectrums. The primary challenge is generating English and Indonesian dense vector representation, a.k.a. embeddings that share a single vector space. Through trial and error, the research proposes using the Universal Sentence Encoder (USE) model to construct bilingual embeddings and FAISS to index the bilingual dataset. The comparison between query vectors and index vectors is done using two approaches: the heuristic comparison with Euclidian distance and a clustering algorithm, Approximate Nearest Neighbors (ANN). The system is tested with four different semantic granularities, two text granularities, and evaluation metrics with a cutoff value of k={2,10}. Four semantic granularities used are highly similar or near duplicate, Semantic Entailment (SE), Topically Related (TR), and Out of Topic (OOT), while the text granularities take on the sentence and paragraph levels. The experimental results demonstrate that the proposed system successfully ranks similar texts in different languages within the top ten. It has been proven by the highest F1@2 score of 0.96 for the near duplicate category on the sentence level. Unlike the near-duplicate category, the highest F1 scores of 0.77 and 0.89 are shown by the SE and TR categories, respectively. The experiment results also show a high correlation between text and semantic granularity.**

***Index Terms*—Textual Similarity Detection, Universal Sentence Encoder (USE), Facebook AI Similarity Search (FAISS)**

Received: Feb. 09, 2024; received in revised form: July 17, 2024; accepted: July 22, 2024; available online: Sep. 09, 2024.
*Corresponding Author

I. Introduction

TEXTUAL Similarity Detection (TSD) has been closely related to Information Retrieval (IR), text retrieval, Plagiarism Detection system (PDS), and Text Reuse Detection (TRD) which fall within the area of Natural Language Processing (NLP). Tracing down research and applications on TSD, it has its root in a detection algorithm proposed by Ottenstein [1] in 1976 which originally designed to identify similarity of source codes written in Formula Translation (FORTRAN). TSD in natural language is initialized by the previous researchers who develop algorithm to detect copies for sorting and indexing unique documents the electronic library [2].

Alhough these research areas equally detect text similarities, their similarity spectrum vary greatly [3]. For example, IR system attempts to find relevant information based on user queries [4], focusing on the topic level similarities [5]. In contrast, PDS and TRD seek similarities at both lexical and semantic levels ranging from phrases, sentences, to entire passages [3]. While TRD and PDS methods are almost similar, TRD focuses on finding illegal text reuse, whereas PDS addresses both legal and illegal similar texts. From the researchers' perspective, the term 'textual similarity detection' can be used interchangeably with TRD/PDS.

Designing an effective TSD system involves addressing challenges similar to those in PDS and text

retrieval that lie on two key points: how to learn the text representation and to model the relevance or similarity matching [4]. The text representation applied in earlier TSD systems has been dominated by Bag-of-Words (BoW) model [6] which matches query vectors and the relevant documents on the basis of lexical level. Though the BoW model, a.k.a sparse vector representation, remains of fundamental importance in recent neural-based IR models, its main drawback lies in the fact that it matches the exact terms. If it matches the related words such as synonyms or homonyms, it needs multi-stage retrieval architecture by adding query expansion, term importance prediction or/and reranker [4, 7].

The drawbacks of sparse vector-based text representation has been overcome by the dense vector representation introduced by previous researchers [8] which is able to match a query to its related terms in context. Thus, finding semantically similar sentences or passages in a monolingual text presents no more meaningful challenges. However, the availability of a large number of electronic texts on the web that is accessible openly allows for the content duplication in different languages and plagiarism. In academic field, it is required to check the originality of one's writing to avoid plagiarism. When monolingual TSD or IR systems fail to retrieve similar texts written in different language, a bilingual TSD/IR is needed.

The research focuses on finding similar bilingual texts on a wide range of similarity spectrums, i.e., topically related to lexically and semantically duplicate texts in Indonesian and English. The main challenge lies in generating English and Indonesian dense vector representation, a.k.a. word and sentence embeddings that put semantically related words/sentences in both languages in the same vector space and have the closer dense vector values. To address this challenge, the researchers propose using dense vectors to retrieve similar texts written in English for an Indonesian text and vice versa. The contribution of the research is set on using Universal Sentence Encoder (USE) and Facebook AI Similarity Search (FAISS) for bilingual similar text retrieval in Indonesian-English.

A. Related Works

Though textual similarity detection is very often applied for information as well as text retrieval [3, 4], it frequently becomes the backbone of more complex applications. For instance, previous researchers [9] utilize TSD outputs to match user questions with those in the database in their open-domain Question-Answering system. Meanwhile, another research [10] employs it in a chatbot that assist students asking questions related to faculties, activities, exams, and admission. Lexical or semantic TSD systems are also applied for detecting text reuse [5, 11], natural language-based textual plagiarism [2, 3, 12], and code plagiarism in programming assignments [13] . In distant learning, TSD is very applicable for building an automatic grading system [14] or essay scoring system [15]. Then, in e-commerce field, recommender systems relying on TSD are proposed [12, 16].

The earlier building blocks of TSD systems comprise two salient stages: text retrieval and text analysis [2]. In text retrieval stage, Vector Space Model (VSM) based on BoW model has dominated the text representation method and technique [2]. Among other term weighting methods, the classic TF-IDF has been predominantly implemented for constructing a sparse vector representation [4]. This sparse retrieval has been a robust method for matching queries to the documents in the lexical and syntactic structure.

The improvement of machine learning, specifically deep learning, has enabled dense retrieval, i.e., retrieval model that employs dense vectors or embeddings to represent texts to capture semantic meaning of input texts [4, 17]. The tools and algorithms for generating dense vectors commonly used in text retrieval are Bidirectional Encoder Representations from Transformers (BERT) [18], Word2Vec [8], Global Vectors (GloVe) [19], and FastText [20] which also provide a means of pretrained words as well as sentence embeddings from the open source corpora [21].

The emergence of transfer learning and the pretrained embeddings leads a path to construct cross-lingual as well as multilingual embeddings which becomes a key facilitator in developing bilingual or multilingual TSD. Recently, there are two salient approaches in constructing bilingual and multilingual embeddings [21]. The first approach generates embeddings directly from sentences or document-aligned bilingual corpora [22, 23]. In contrast, the second approach applies transfer learning functions that will project the vectors from the embeddings from one language that of another one [21]. A significant accomplishment in constructing the multilingual embedding is achieved by Google scientists [24] who propose USE. This multilingual USE embeds texts from 16 languages into a single shared semantic space with a multi-task dual encoder [24].

The sparse retrieval-based TSD systems commonly employ an inverted index which mainly comprise a dictionary and a collection of posting list to accelerate the matching process [17]. The dictionary enlists all terms found in the corpora, and each posting list saves information on the terms such as document identifiers where the terms are found and their frequency or

vector values are shown [2]. Such indexing scheme presents significant challenges when it is applied for dense retrieval-based bilingual TSD. In meeting these challenges, previous researchers [25] have proposed SPANN which is a memory-disk hybrid vector indexing and search system based on inverted index methodology. An open-source indexing and search system known as FAISS for dense retrieval is proposed by previous research [26]. Both SPANN and FAISS can be used to index a large number of text representations.

There is a handful work on Indonesian-English TSD based on dense vectors as its text representation. The first example generates dense vectors from the aligned Indonesian-English sentence pairs using Word2Vec. The matching is done by aligning the dense vectors in Indonesian to those in English by computing their canonical correlation analysis [27]. Like the first example, the second example constructs word embeddings with Word2Vec to build a monolingual TSD system. However, the matching is done by directly measuring the vector similarity with cosine similarity metric [28]. Back to bilingual TSD, the third example captures the semantic similarity by utilizing Latent Semantic Analysis (LSA) and Linear Vector Quantization (LVQ) to optimize its performance [29]. Different from these Indonesian-English TSD, the proposed method uses USE in generating the dense vector, FAISS for its indexing, and similarity metrics in making text comparison.

II. Research Method

The research starts with methods used for building the training and testing dataset. Then, it describes the system architecture and its components, concluding with the evaluation metrices.

A. Dataset Building

After some trials and errors in designing the datasets, the researchers considers the approach of constructing the embeddings directly from the training dataset as suggested in previous research [22, 23]. Thus, the researchers constructs pairs of Indonesian-English sentences and paragraphs. The detailed process of constructing the dataset construction is described in the following paragraphs.

To begin with, the researchers decide to use Wikipedia as the source of the dataset due to its availabillity of articles on the same topics in both Indonesian and English. A web scrapping code is written in Python to grab Wikipedia texts. As constraints for the web grabbing, articles in Indonesian which have their equivalences in English are grabbed by checking the English version link that is provided in the left pane on each Wikipedia's article. The web scrapping yields numerous article pairs. However, only some parts of their contents show corresponding semantic relations in Indonesian and English. Therefore, the researchers manually parallelize the Indonesian text with its counterpart in English.

The corpus parallelization process is done by students with good English proficiency under the following instructions:

1) They are not allowed to change, rewrite, or translate the content. The aim is to preserve the natural content of the article.
2) They need to reorder the sentences in a paragraph, so each Indonesian paragraph matches the English paragraph in the number of sentences, ensuring that semantically equivalent sentences are in the same order.
3) They are allowed to delete Indonesian sentences in a paragraph if its English counterpart cannot be found in other paragraphs in the entire article.
4) They can insert a sentence found in another paragraph if that sentence is an equivalence of its counterpart sentence that is missing in one specific paragraph checked on.
5) If the English counterpart has two paragraphs, while the Indonesian consists only one paragraph, they are allowed to split Indonesian paragraph into two paragraphs to maintain the order of equivalent sentences.
6) If an English sentence has two Indonesian counterparts, they can merge those sentences into one and adjust the grammatical form accordingly, like a dependent clause preceding or following the main clause.

The example of pairs of raw paragraphs and their reordered ones are displayed in Figs. 1 and 2. It shows the results of the application of the rules described previously. The reordered and aligned paragraphs of each article are then saved in each plain text format which is given an identifier (ID) taking its source language as its main marker. For example, ID001.txt is for an Indonesian article while EN001 is ID001's equivalence in English. The shortest document consists of 3 paragraphs while the longest comprises 19 paragraphs.

The selection of the articles to be included in the work of reordering is also based on their topics that are randomly chosen. If they convey the predefined topics, they will be selected. There are 44 topics that the researchers work on. The list is displayed on Table I. The determination whether an article belongs to one of these topics is also done manually by students who do the reordering. As the article reordering is done

Cite this article as: L. D. Krisnawati, A. W. Mahastama, S. C. Haw, K. W. Ng, and P. Naveen, "Indonesian-English Textual Similarity Detection Using Universal Sentence Encoder (USE) and Facebook AI Similarity Search (FAISS)", CommIT Journal 18(2), 183–195, 2024.

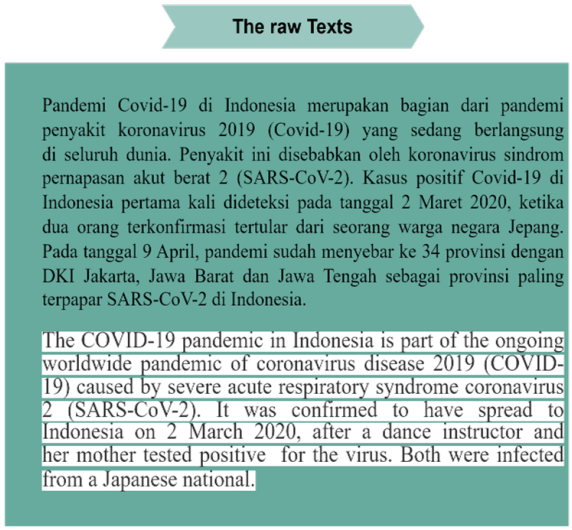

Fig. 1. Raw texts in Indonesian and English.

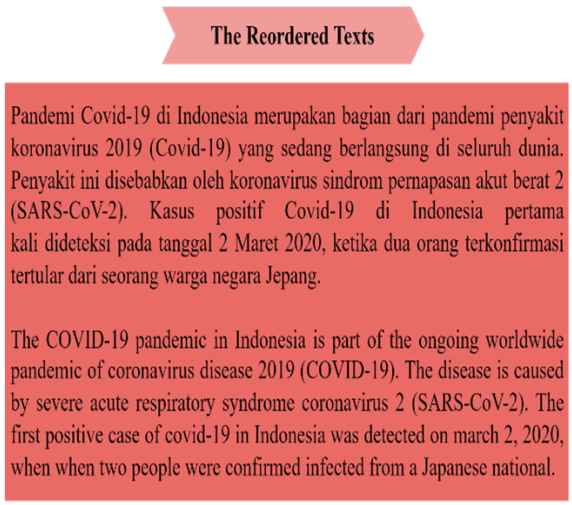

Fig. 2. Ordered texts in Indonesian and English.

manually, it is laborious and time consuming. Considering time span and budget constraints of the research, it stops after having 116 documents as training and testing data. Articles having different topics from those in training data are chosen as Out of Topic (OOT) documents. The preprocessing of these documents is described in the following subsection.

B. System Architecture

The architecture of the proposed semantic text retrieval is described. Its steps shown in Fig. 3 follows the semantic search system. Figure 3 shows that the process of developing the TSD system starts with the

TABLE I
LIST OF TOPICS COVERED IN THE DATASET.

No.	Topics	No.	Topics
01	Pandemi COVID-19 di Indonesia	23	Muria
02	ChatGPT	24	Lawu
03	Keraton	25	Gunung Agung
04	Malioboro	26	Merapi
05	RAtu Boko	27	Bali
06	Candi Prambanan	28	Bahasa Inggris
07	Pelangi	29	Children
08	Tugu Jogja	30	Indonesia
09	Gerhana matahari	31	Jumat Agung
10	Saturnus	32	Kamis Putih
11	Lawang Sewu	33	Komodo
12	Telegram	34	Kotonoha
13	Venus	35	Minggu Palma
14	Ikan Hiu	36	Nato
15	Pura	37	Paskah
16	Kelapa	38	Perang Dunia
17	Robotika	39	Puasa dan Pantang
18	Weathering with You	40	Rendang
19	Pattimura	41	Rusia
20	Singapura	42	Soeharto
21	Your Name	43	Suzume
22	Merbabu	44	Wayang

web scrapping from Wikipedia portal as described earlier. The results of web scrapping are processed to ensure that they contain the same number of paragraphs and sentences in the same order. These plain textdocuments are then preprocessed. The next step is to construct the dense vector which utilizes the Multi-Lingual Universal Sentence Encoder (MUSE) model provided by Google Scholar as an open source model. The dense vectors of these documents are then indexed using FAISS library. FAISS library is employed to match or compare the query of dense vectors with those documents in the dataset.

C. Preprocessing Phase

The reordered documents described previously are saved in separate subfolders of the dataset folder according to its language clusters. The text preprocessing is applied for each document in each cluster. Overall, the researchers implement two stages of preprocessing texts only. They are text normalization and text segmentation. Two processes of text normalization (case folding and punctuation elimination) are implemented. In case folding, all capital letters are transformed into lowercase letter. It is decided to eliminate all punctuations except those that are commonly used to deliminate sentences or to mark a part of sentences. Thus, the punctuations kept are period, comma, exclamation marks, question marks as well as new line as the paragraph delimiter [. , ! ? \n].

After a document has been normalized, the segmentation in the level of paragraph is done by using the new line symbol \n. Headings and subheadings are treated as a separate paragraph with a reason that they

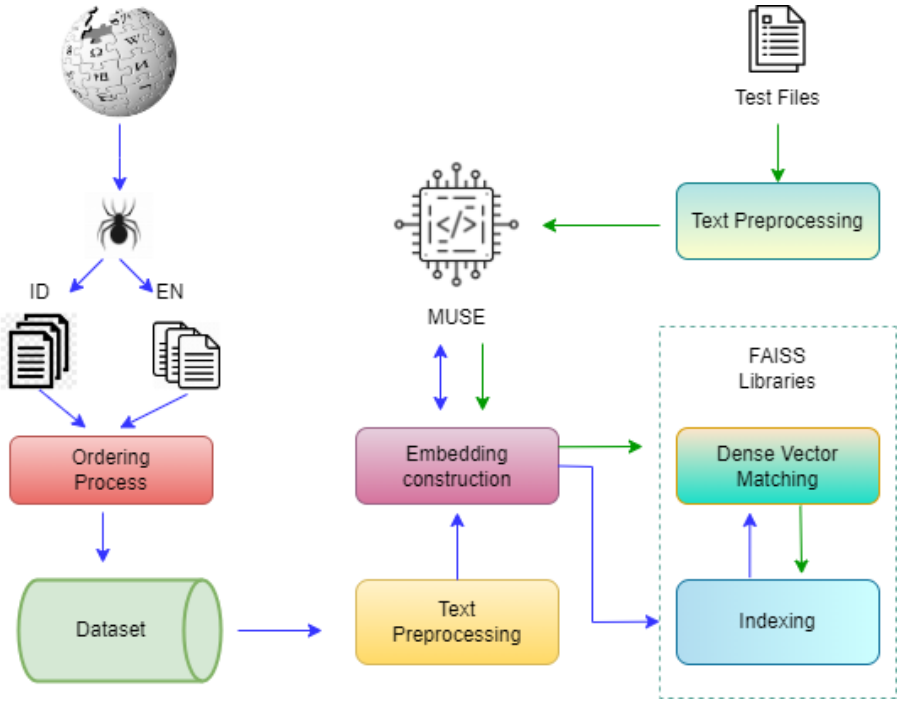

Fig. 3. The architecture of the proposed system.

have no sentence delimiter. If they are merged into their succeeding paragraph, it will be a part of the first sentence in that paragraph. The identification of each segmented paragraph was done by using the language identifier (ID/EN) followed by five digits in which the first three digits are allocated to their index number of processing iteration, and the last two digits show the index of paragraph in a currently processed document.

Sentence segmentation is applied to the segmented paragraphs using the Natural Language Toolkit (NLTK) tokenize package and the *sent_tokenize()* method. In this case, the headings and subheadings are treated as single sentences. The process of assigning identifier for each segmented sentence follows the paragraphs. The distinction lies in the last two digits which are reserved for the index of each sentence within a document. Both the segmented sentences and paragraphs are saved in a separate CSV formatted document.

D. Bilingual Embedding Construction

Initially, the research intends to use pretrained word embeddings from a repository openly provided by the Language Technology Group, at the University Oslo (http://vectors.nlpl.eu/repository/). This repository provides not only multilingual word embeddings, but also detail descrition of the model, hyper-parameters and corpora used to train them. It also provides two models of Indonesian word embeddings trained on different corpora. However, using English and Indonesian word embeddings from this repository presents a twofold problem, i.e., system crash due to the hardware limitations and the inherent issues of separately trained embedding models. The hardware constraint can be readily addressed with cloud computing, but the second issue poses a significant challenge.

The main issue of the second problem is that the Indonesian embeddings are trained using different models than English embeddings. Theoretically, the vector space spanned by each embedding model is distinct [21]. Consequently, the Indonesian and English vectors, having been trained from different models, cannot be directly compared because they do not reside in the same vector space. Previous research [21] has encountered this issue and proposed a solution by mapping these disparate vectors to a single space to achieve comparability. In response, the researchers seek an embedding modell capable of projecting multilingual embeddings in a single vector space, and the researchers identify the USE model as a suitable option.

USE, an abbreviation of Universal Sentence Encoder, is a model for encoding sentences into embedding vectors that specifically target transfer learning to other NLP tasks proposed by Google research team [30]. USE makes two models namely Trans-

```
import tensorflow_hub as hub

Module_url = 'https://tfhub.dev/google/universal-
sentence-encoder-multilingual/3'
embed = hub.load(module_url)
```

Fig. 4. A snippet on how to call MUSE.

```
import faiss

index = faiss.IndexFlatL2(embed_dim)
index_i = faiss.IndexIDMap(index)
index_i.add_with_ids(embeddings, lg_ids)
```

Fig. 5. The simplified snippet for creating a flat index in indexing with Facebook AI Similarity Search (FAISS).

former and Deep Average Network (DAN) [30]. There are several versions of USE. The one that generates bilingual embeddings in one vector space is MUSE. MUSE is a new member of USE which is a multilingual sentence embedding model based on Convolution Neural Network (CNN) and Transformer architecture. The followings are the summary information on MUSE [24]:

1) CNN encoder uses 2 CNN layers with filter width of [1, 2, 3, 5] and 256 filters per width.
2) Transformer encoder employs 6 transformer layers, with 8 of attentions heads, 512 of hidden size, and 2048 of filter size.
3) The models are implemented in Tensor-Flow and made available on TensorFlow Hub.
4) The models embed 16 languages into a single and shared semantic space using a multi-task trained dual encoder. Unluckily, Indonesian is unavailable.
5) It serves three tasks: question-answering task, translation ranking task, and natural language inference task [24].

MUSE model is publicly available and downloadable to the local PC or server for offline usage, but it also can be directly called from its URL. Figure 4 demonstrates the call of MUSE. Firstly, the MUSE model is called from tfhub.dev library. Secondly, the researchers need to load it from the tensorFlow-hub module. A text can be passed on as a sole parameter of this module since it automatically creates an embedding with 512 dimensions. The code snippet in Fig. 4 is simplified due to the limited space here.

E. Indexing with Facebook AI Similarity Search (FAISS)

In term-based retrieval system, the most popular indexing scheme is the inverted index due to its efficiency and simplicity [17]. However, its efficiency becomes a question when it is applied to index dense vectors. Thus, the researchers turn to the available open-source library for dense vector indexing. The researchers come upon FAISS which provides several indexing algorithms, capable of indexing a large number of document embeddings. Among these indexing algorithms, the researchers apply two indices: indexflatL2 and indexIVFFlat.

IndexflatL2 belongs to flat index class which has two variants of metrics for calculating the vector distances. It is coined as a flat index because it neither modifies any vectors that are stored in it, nor trains them [31]. Given a set of dense vectors from a query text, IndexflatL2 simply compares them to every other full-size vector in the index. To implement indexflatL2, the researchers need to import FAISS, the dense vector dimensionality, and data/documents. This index needs a single parameter, i.e., the dimensionality of the dense vector. Figure 5 shows the simplified code snippet on how to implement indexflatL2.

Figure 5 shows that after initializing the indexFlatL2 with the vector dimension, it needs to be encapsulated to another index with $IndexIDMap()$ module. It is done to avoid crash since indexFlatL2 does not support the $add\_with\_ids()$ module. After the encapsulation, the dense vector along with their identifiers can be stored in the index using the $add\_with\_ids()$ module. The search or comparison between query vectors and those in the index in FlatL2 is measured by L2-norm Euclidean distance [31].

IndexIVFFlat is an optimization of flat index by partitioning it into Voronoi cells [31]. The partitioning is done by clustering the vector embeddings. Thus, a centroid is computed for each cell or cluster. Given a set of dense query vectors, indexIVFFlat compares the query vectors with the centroids. Then, the search is restricted to the partitions whose centroids have minimum distances from the query vectors. The comparison is done by applying ANN [17, 26]. To implement indexIVFFlat, the researchers need to call indexflat to quantize the newly created index. In this case, the researchers use the indexflatIP variant which employs Inner Product (IP) instead of Euclidian distance. As indexIVFFlat makes use of clustering, it needs a parameter that states clearly the number of cells or clusters in addition to the vector dimension. The vector embeddings are needed to be trained in this index. Lastly, the researchers add the embeddings and their

```
import faiss

quantizer = faiss.IndexFlatIP(embed_dim)
index = faiss.IndexIVFFlat(quantizer, embed_dim, ncells)
index.train(embeddings)
index.add_with_ids(embeddings, lg_ids)
```

Fig. 6. A simplified snippet for creating indexIVFFlat.

identifiers to the index by calling the $add\_with\_ids()$ method. In the bilingual setting, this process that needs to iterate is the addition of vector embeddings and their IDs in the index. Figure 6 shows the simplified way of creating index with FAISS indexIVFFlat.

F. Evaluation Metrics

In sparse retrieval, the popular metrics for measuring its performance are Precision, Recall, and F1 Score. Since the highly similar texts are always ranked at the top of TSD system outputs, the researchers decide to use the cut-off metrics proposed by previous research [4]. These are the commonly used metrics, however their computations stop after k points. Thus, Precision@k calculates the proportion of texts weighted highly similar at the top k and is computable by the Eq. (1). Q denotes a set of queries while q represents an individual query. k is a is a cut-off point, while $\#ret_{q,k}$ stands for the retrieved documents whose similarity are ranked on the top k.

$$Prec@k = \frac{1}{Q}\sum_{q=1}^{Q}\frac{\#ret_{q,k}}{k}. \quad (1)$$

Like Precision@k, Recall@k computes the proportion of the texts weighted highly similar by the TSD system among those labeled having highly similarity. It is computed by Eq. (2) where the number of retrieved documents, $ret_{q,k}$ is divided by the number of relevant-labelled documents to query (rel_q). Then, F1@k as shown in Eq. (3) is a harmonic means of Precision@k and Recall@k as well.

$$Rec@k = \frac{1}{Q}\sum_{q=1}^{Q}\frac{\#ret_{q,k}}{rel_q}, \quad (2)$$

$$F1@k = \frac{1}{Q}\sum_{q=1}^{Q}\frac{2 * Prec@k * Rec@k}{Prec@k + Rec@k}. \quad (3)$$

III. Results and Discussion

The experiment and evaluation are conducted with the dataset with described building process in previous section. However, it has not described the building process of the test documents. Hence, this section presents the testing document creation and testing scenario, followed by the experiment results and its discussion.

A. Testing Document Creation

In creating the test documents, the researchers hire three students from Informatics and English Departments. Their main task is to rewrite some documents in Indonesia having semantic similarity with those in the dataset. They are given the list of topics listed in Table I and the document IDs belong to those topics. They are asked to choose some topics and documents belong to each topic they have selected. To accomplish their tasks, they are provided the following instructions:

1) Read the selected documents thoroughly.
2) Choose minimally three and maximally five paragraphs from a particular document.
3) Rewrite the paragraphs by paraphrasing it and keep in mind that the rewritten texts need to have high degree of semantic similarity from their sources.
4) Label the document with test document ID followed by the source document ID. For example, the source document ID is id005. Then the ID for the test document is tid005.
5) Allow them to rewrite a paragraph so that it has different number of sentences from its source.
6) Put each rewritten paragraph to Google Translate to translate it into English.
7) Save the translated version as the English test documents that the naming process takes the same step as in number 4, e.g., ten005.
8) Proofread the translation result. If the Google Translate output contains grammatical mistakes, the students from the English Department should rectify it.
9) Choose articles from Wikipedia manually whose topics are not on the lists. For example, there are topics on Malioboro and Keraton in the list, so finding articles on Javanese culinary or culture will fit this criteria. These articles are included as the OOT test set.

The queries or test texts are formulated from the rewritten documents along with their translation. These documents undergo preprocessing. Some documents are randomly selected and segmented into paragraphs. The rest of the documents are segmented into sentences. Both the rewritten and translated texts are saved in a comma-separated values (.csv) and plain text (.txt). Thus, the researchers have four pairs of query categories, i.e., Indonesian queries in the level of paragraph, English queries in the form of paragraphs,

TABLE II
THE STATISTIC DATA ON THE DATASET.

Granularity	Number of Documents		
	Indonesian	English	Total
Dataset to Index			
- Sentences	1,917	1,917	3,834
- Paragraphs	558	558	1,116
Test set			
- Sentences	98	98	196
- Paragraphs	28	28	56
- Out of Topic (OOT) texts	11	11	22

TABLE III
THE RETRIEVAL RESULTS FOR HIGHLY SIMILAR TEXT.

Granularity	Index Algorithm	F1@2	Prec@10	Rec@10	F1@10
Paragraph	FlatL2	0.91	**0.20**	**1.00**	**0.33**
	The Researchers-10	0.84	0.18	0.91	0.30
	IVF-15	0.84	0.17	0.86	0.28
	IVF-20	0.95	**0.20**	0.98	**0.33**
Sentences	FlatL2	**0.96**	**0.20**	**1.00**	**0.33**
	IVF-10	0.92	0.19	0.98	0.32
	IVF-15	0.85	0.18	0.89	0.29
	IVF-20	0.86	0.17	0.86	0.29

Note: IVF= Inverted Index Flat

Indonesian and English queries in the form of complete, and well form sentences. The statistic data on the datasets included the test set can be seen in Table II.

B. Experiment Results

The test dataset as queries are inputted into the proposed model. In iteration, the dense vectors of each query are computed using MUSE, then they are fed to FAISS to be compared with the dense vectors of the indexed texts. The researchers experiment indexflatL2 and indexIVFFlat on the queries, as described on Table II. For indexIVFFlat, the researchers partition the dense vectors into 10, 15, and 20 clusters and use the Inner Product (IP) for comparison among the dense vectors. The evaluation metrices described previously are applied, and the cutoff values of k are set up to 2 and 10. The results of this experiment are displayed on Table III.

At first, the researchers set up the cutoff value k to be equal to 2 (k=2) with a reason that the researchers need to check whether the highly similar text from the query in a different language can be matched and retrieved. However, with only two texts in Indonesian and English labeled as having high similarity for each query, the computation of the confusion matrix for precision, recall and F1 score yields exactly the same rate when k is set up to 2. Therefore, only the F1 score with cutoff value 2 is displayed in Table III.

In all experiment settings, the F1@2 scores are relatively high with the highest rate of 0.92 and the lowest rate of 0.84. These rates also show a pattern where the finer granularity, sentences, achieves higher F1 scores than paragraphs with an exception on the IVF-20 setting only. With such rate, the researchers think that the high score of F1@2 is bias due to the same values between k and the number of labeled matches. For that reason, it will be interesting to check the recall rate if the cutoff value k is set to 10. Although the precision rates drop significantly in all experiment settings, the recall rates for indexFlatL2 both for sentence and paragraph levels achieve the maximal rate, 1.0.

In this experiment, FlatL2 outperforms IVFFlat because FlatL2 compares the dense vectors exhaustively. For example, each set of dense vectors in the index will be compared with those in the query. The Euclidian Distance or L2 is used to measure the distance between these vectors. As a result, index FlatL2 provides accurate output. However, it suffers in high computation time. In contrast, IVFFlat tries to reduce the computation time by applying clustering using Voronoi Diagram concept. It is a little bit faster than FlatL2. However, its search quality experimented with 10, 15, and 20 number of clusters is still below the search quality of FlatL2.

As the researchers scrutinize the top ten ranked texts for each query, the researchers realize that the low scores of Precision@10 leading to low F1@10. Hence, the researchers are unable to judge the low performance of the proposed method. The reason is that some texts ranked on the top ten have semantic relation with the query but they are not labeled as a match due to the annotators' ignorance on the existence of such texts in the dataset. Thus, measuring the performance on highly similar or near-duplicate texts based on the label only will diminish the real performance of the proposed model.

To overcome the drawback of such evaluation computation, the researchers design the semi manual evaluation by measuring two other categories: the semantic entailment and topically related texts. The researchers defines the following criteria:

1) Highly similar or near duplicate: the source texts are the queries that students rewrite. They are labelled as the match and algorithmically computed. Their computation results are displayed in Table III.
2) Semantic Entailment (SE): the sentences or paragraphs share approximately 20–50% terms in the query, and/or the topic of the query entails the topic of texts ranked on top ten or vice versa.

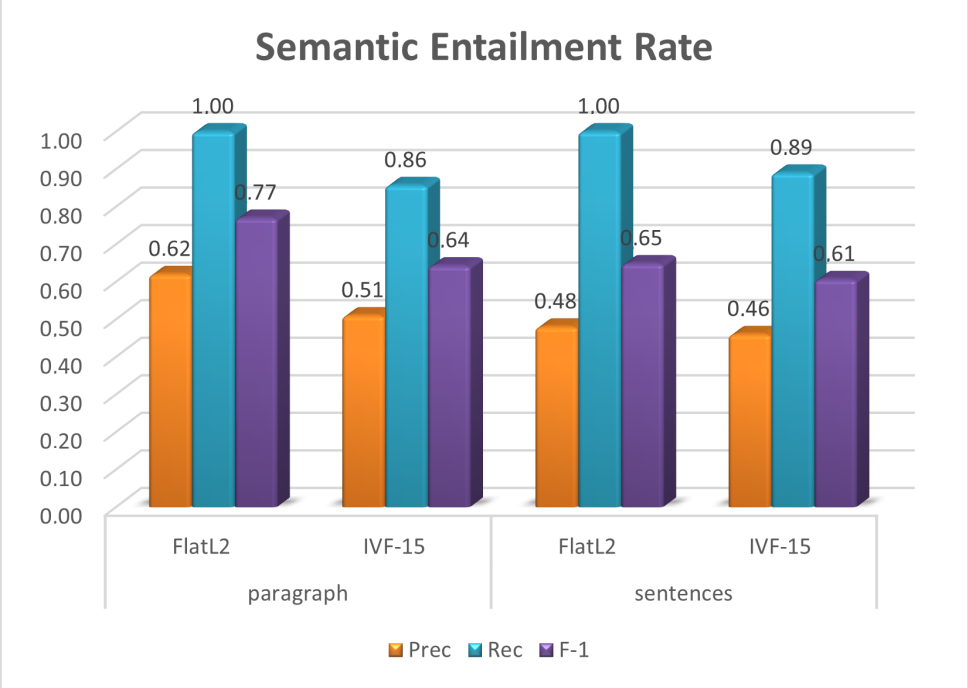

Fig. 7. Precision, recall, and F1 scores for Semantic Entailment (SE).

3) Topically Related (TR): the sentences or paragraphs share approximately up to 19% semantically related terms in the query and/or the topic is strongly related. For example, the query talks about the Sultan palace, and the text discusses on Sultan's coronation in the palace.
4) OOT: a query is included in OOT when its topic has no relation to all topics listed in Table I. Its precision is judged manually by human readers who read all top ten ranked texts. If they have semantic relation, these texts will be considered as relevant to OOT.

The evaluation metrics for SE and TR categories are semi-manually computed, meaning that the judgement of the top ranked texts considered relevant is done manually by human readers. It is done by reading each top ranked text and decide whether the text belong to one of four categories above based on the requirements of each category. However, they count the number of relevancy and input the evaluation system. Then, the precision score is algorithmically computed. Figure 7 displays the metric rates for the SE category.

The SE concept has a wider spectrum so that it includes the texts in the near duplicate category. Thus, it will automatically increase the Precision@10 rate. As number of the indexed texts falling in this category is unknown and hidden in the dataset, the researchers base the recall computation on the number of relevant-labelled texts as in the highly similar or near duplicate category. In other words, the researchers use their Recall@10 rate to compute the F1@10 score. The same approach is applied in computing the precision@10 rate for TR category. Figure 8 shows the results of evaluation metrices for TR category.

As the similarity spectrum of TR category is coarser and more general, the near-duplicate and SE concepts become its subsets. For this reason, the computation of Precision@10 for TR category includes texts labeled as highly similar and SE. It explains why the Precision@10 score of TR is higher in all experiment settings compared to its former categories.

The experiment on OOT aims to check the vast coverage of the proposed model in retrieving texts having semantic relation. Logically, given texts having topics different from dataset, the system will output non-sense texts. Since this is a semantic search using bilingual dense vectors, the researchers would like to see what kind of texts ranked on the top 10. The semantic relation in OOT context refers to lexical as well as contextual relations. The lexical relation comprises synonyms, antonyms, homonyms, hypernyms, and polysemy. Meanwhile, the contextual relation includes SE, association, and connotation as well. These are criteria to judge the relevancy of top 10 ranked text outputs to OOT. The Precision@10 achieves 0.89 rate (see Fig. 9) shows that the proposed system is not only

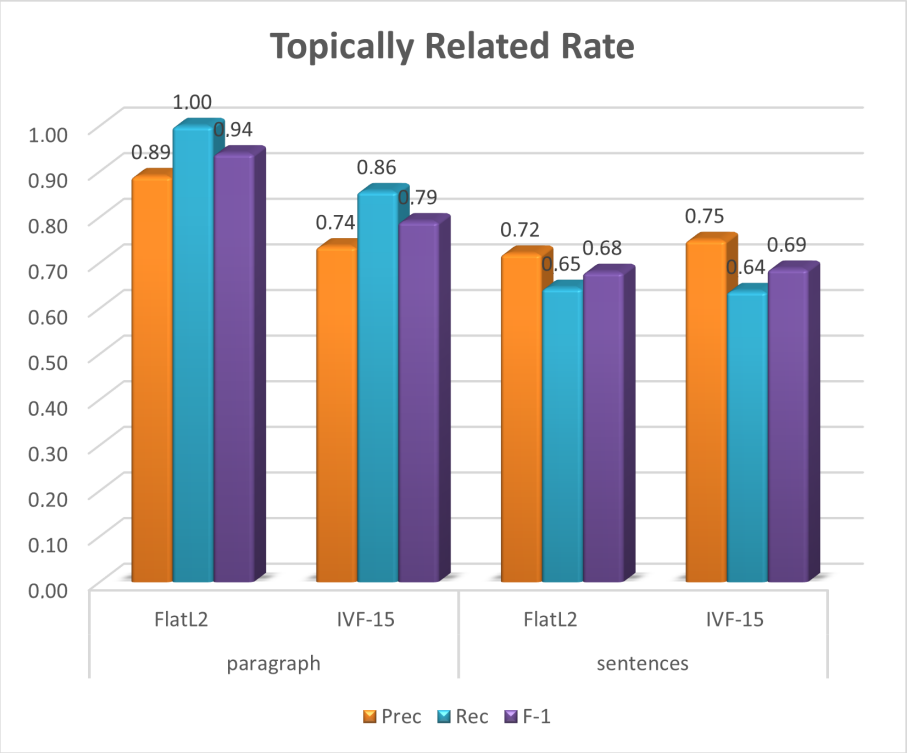

Fig. 8. The experiment results on the Topically Related (TR) category.

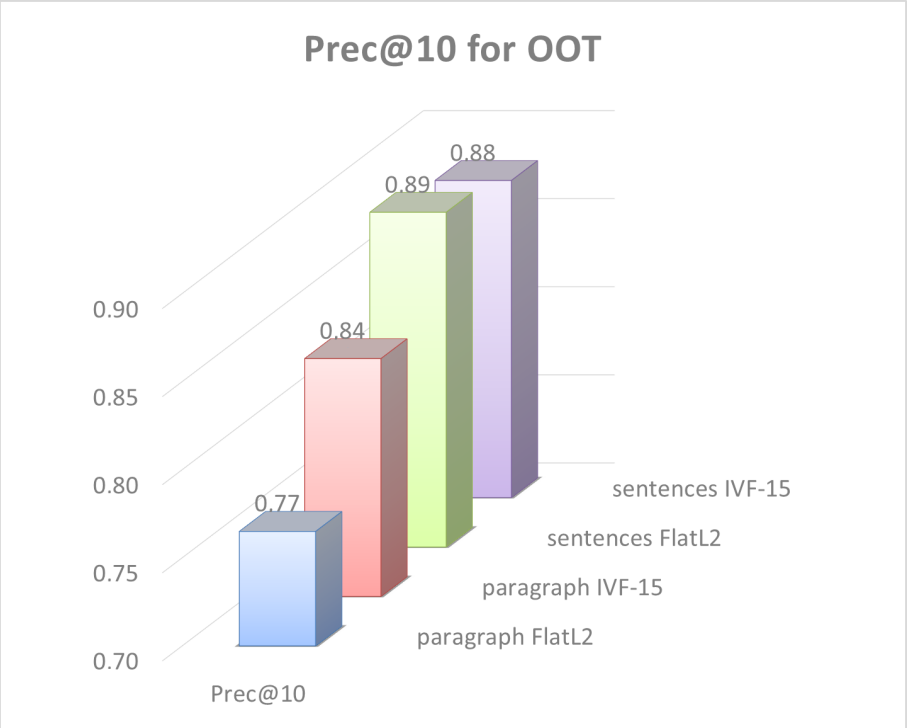

Fig. 9. The precision rate for Out of Topic (OOT) scheme.

capable of identifying such semantic relation only, but also ranking highly texts having semantic relation in different language than the query–Indonesian–English or vice versa.

The results of four experiment settings presented in Table III and Figs. 7–9 show that the high scores for text granularity correlates with the semantic granularities. This is demonstrated by F1 scores that are higher on the fine-grain level, sentences, for the finer semantic granularity–highly similar or near duplicates (cf., Table III). The F1 scores for the coarse-grain level – paragraphs are higher mostly in the setting of coarse-grained semantic topics which are in the SE and TR (cf., Figs. 7 and 8).

IV. Conclusion

Focusing on identifying and detecting semantic similarities between texts in Indonesian and English, the research successfully implements the pretrained USE model to construct dense vectors of the bilingual dataset in a single shared vector space. The indexing of these dense vectors is achieved by devising FAISS, while the comparison between query vectors and those in the index is done by employing two different approaches, i.e. the heuristic comparison with L2-norm Euclidian Distance and a clustering algorithm, ANN.

In evaluating the performance of the proposed system, the cutoff metrices @k used are Precision, Recall, and F1 scores at k = 2, 10. Tested across four different semantic granularities and two text granularities, the experiment results demonstrate a strong correlation between text granularity and semantic granularity. The results are shown by the F1 scores for fine-grained texts which are higher in the setting of fine-grained semantic relations. Conversely, F1 scores for coarse-grained texts are predominantly higher in the experimental setting of coarse-grained semantic topics such as SE and TR settings.

Due to time and budget constraints, the research is limited to the initial module of textual similarity detection, focusing on retrieving bilingual texts having length less than 1,500 words as its training data. The identification, the alignment of the semantically passages, and the computation of similarity percentages for longer texts will be pursued in future work.

Acknowledgement

The research was supported by a grant from UKDW-MMU Joint Research Grant UKDW-RU23, MMUE/230033. The authors are indebted to both institutions which provided a grant to assist with the research

Author Contribution

Writing—original draft, L. D. K.; Methodology and model, L. D. K.; UI design, A. W. M.; Experiment scenario, H. S. C.; Data labeling and experiment evaluation, K. W. N. and P. N. All authors have read and agreed to the published version of the manuscript.

References

[1] K. J. Ottenstein, "An algorithmic approach to the detection and prevention of plagiarism," *ACM SIGCSE Bulletin*, vol. 8, pp. 30–41, 1976.

[2] L. D. Krisnawati, "Plagiarism detection for Indonesian texts," PhD Thesis, Faculty for Languages and Literatures, Ludwig-Maximilian University, 2016. [Online]. Available: https://edoc.ub.uni-muenchen.de/19823/

[3] L. D. Krisnawati, J. F. Lim, and G. Virginia, "Penggunaan pemodelan topik dalam sistem temu kembali dokumen termirip," *Jurnal Linguistik Komputasional*, vol. 6, no. 3, pp. 1–10, 2023.

[4] W. X. Zhao, J. Liu, R. Ren, and J. R. Wen, "Dense text retrieval based on pretrained language models: A survey," *ACM Transactions on Information Systems*, vol. 42, no. 2, pp. 1–60, 2024.

[5] N. C. Haryanto, L. D. Krisnawati, and A. R. Chrismanto, "Temu kembali dokumen sumber rujukan dalam sistem daur ulang teks," *Jurnal Teknologi dan Sistem Komputer*, vol. 8, no. 2, pp. 140–149, 2020.

[6] J. Lin, "A proposed conceptual framework for a representational approach to information retrieval," *ACM SIGIR Forum*, vol. 55, no. 2, pp. 1–29, 2022.

[7] A. Yates, R. Nogueira, and J. Lin, "Pretrained transformers for text ranking: BERT and beyond," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. Virtual Event Israel: Association for Computing Machinery, March 8–12, 2021, pp. 1154–1156.

[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, 2013, pp. 3111–3119.

[9] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. T. Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Online: Association for Computational Linguistics, Nov. 2020, pp. 6769–6781.

[10] A. Mundher, K. Khater, and L. M. Ganeem, "Adopting text similarity methods and cloud com-

puting to build a college chatbot model," *Journal of Education and Science*, vol. 30, no. 1, pp. 117–125, 2021.

[11] L. Gienapp, W. Kircheis, B. Sievers, B. Stein, and M. Potthast, "A large dataset of scientific text reuse in open-access publications," *Scientific Data*, vol. 10, pp. 1–11, 2023.

[12] N. Ghasemi and S. Momtazi, "Neural text similarity of user reviews for improving collaborative filtering recommender systems," *Electrononic Commerce Research and Appllications*, vol. 45, 2021.

[13] O. Karnalim, S. Budi, H. Toba, and M. Joy, "Source code plagiarism detection in academia with information retrieval: Dataset and the observation," *Informatics in Education*, vol. 18, no. 2, pp. 321–344, 2019.

[14] M. Chen and Y. Dong, "Design of exercise grading system based on text similarity computing," *Mobile Information Systems*, vol. 2022, no. 1, pp. 1–7, 2022.

[15] M. R. R. Susanto, H. Thamrin, and N. A. Verdikha, "Performance of text similarity algorithms for essay answer scoring in online examinations," *Jurnal Teknologi Informasi (JUTIF)*, vol. 4, no. 6, pp. 1515 – 1521, 2023.

[16] M. Oppermann, R. Kincaid, and T. Munzner, "VizCommender: Computing text-based similarity in visualization repositories for content-based recommendations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, p. 495–505, 2021.

[17] J. Guo, Y. Cai, Y. Fan, F. Sun, R. Zhang, and X. Cheng, "Semantic models for the first-stage retrieval: A comprehensive review," *ACM Transactions on Information Systems (TOIS)*, vol. 40, no. 4, pp. 1–42, 2022.

[18] P. Nie, Y. Zhang, X. Geng, A. Ramamurthy, L. Song, and D. Jiang, "DC-BERT: Decoupling question and document for efficient contextual encoding," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Virtual Event China, July 25–30, 2020, pp. 1829–1832.

[19] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," 2014.

[20] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[21] J. P. Sanjanasri, V. K. Menon, K. P. Soman, S. Rajendran, and A. Wolk, "Generation of cross-lingual word vectors for low-resourced languages using deep learning and topological metrics in a data-efficient way," *Electronics*, vol. 10, no. 12, pp. 1–23, 2021.

[22] S. Ruder, I. Vulić, and A. Søgaard, "A survey of cross-lingual word embedding models," *Journal of Artificial Intelligence Research*, vol. 65, pp. 569–630, 2019.

[23] M. Niyogi, K. Ghosh, and A. Bhattacharya, "Learning multilingual embeddings for cross-lingual information retrieval in the presence of topically aligned corpora," 2018. [Online]. Available: https://arxiv.org/abs/1804.04475

[24] Y. Yang, D. Cer, A. Ahmad, M. Guo, J. Law, N. Constant, G. H. Abrego, S. Yuan, C. Tar, Y. H. Sung, B. Strope, and R. Kurzweil, "Multilingual universal sentence encoder for semantic retrieval," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Online: Association for Computational Linguistics, Jul. 2020, pp. 87–94.

[25] Q. Chen, B. Zhao, H. Wang, M. Li, C. Liu, Z. Li, M. Yang, and J. Wang, "SPANN: Highly-efficient billion-scale approximate nearest neighborhood search," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5199–5212, 2021.

[26] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[27] Y. Arifin, S. M. Isa, L. A. Wulandari, and E. Abdurachman, "Developing a bilingual model of word embedding for detecting Indonesian English plagiarism," *Journal of Theoretical and Applied Information Technology*, vol. 99, no. 17, pp. 4388–4348, 2021.

[28] N. R. Ramadhanti and S. Mariyah, "Document similarity detection using Indonesian language Word2Vec model," in *2019 3rd International Conference on Informatics and Computational Sciences (ICICoS)*. Semarang, Indonesia: IEEE, Oct. 29–30, 2019.

[29] A. A. P. Ratna, P. D. Purnamasari, B. A. Adhi, F. A. Ekadiyanto, M. Salman, M. Mardiyah, and D. J. Winata, "Cross-language plagiarism detection system using latent semantic analysis and learning vector quantization," *Algorithms*, vol. 10, no. 2, pp. 1–14, 2018.

[30] D. Cer, Y. Yang, S. Y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, B. Strope, and R. Kurzweil,

"Universal sentence encoder for English," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Brussel, Belgium, Nov. 2018, pp. 169–174.

[31] J. Briggs, "FAISS: The missing manual." [Online]. Available: https://www.pinecone.io/learn/series/faiss